International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V73I7P124

Volume 73 Issue 7, 308-317, July 2025
© 2025 Seventh Sense Research Group®

Original Article

An Efficient Lightweight Cryptography Tiny Hash Function for Data Security in IoT Applications

Mangalampalli Kameswara Subrahmanyam[1], Kunjam Nageswara Rao[2]

[1,2]*Department of Computer Science and System Engineering, Andhra University, Visakhapatnam, Andhra Pradesh, India.*

[1]*Corresponding Author : kameswarasubrahmanyam@gmail.com*

Received: 11 April 2025 Revised: 25 June 2025 Accepted: 30 June 2025 Published: 30 July 2025

Abstract - *Safeguarding data is vital in today's rapidly evolving technological landscape. Instant encryption and transmission are imperative to prevent unauthorized access by potential attackers. Lightweight cryptography hash functions are in high demand since traditional cryptographic techniques are unsuited for Internet of Things (IoT) applications with limited resources. This research presents the development of a new lightweight cryptographic hash function, i.e., Tiny Hash Function (THF). The proposed design accepts an input message of any length and employs processing the message blocks, compression functions and round calculations, substitution and linear diffusion layers to produce 64H/128H/256H hex hash values. The strength of the proposed algorithm is evaluated according to Shannon's property of "confusion" and "diffusion" and is recorded as 70.2%, demonstrating high sensitivity to input message changes. THF has undergone extensive testing to ensure its security, concentrating on three essential lightweight requirements: benchmarking against other hash functions for collision resistance, preimage resistance, and second preimage resistance. The suggested hash function is essential for protecting sensitive data in cybersecurity, finance transactions and data management sectors.*

Keywords - *Confusion, Diffusion, Hash function, IoT, Light-weight cryptography, Security.*

1. Introduction

The Internet of Things (IoT) now encompasses most modern-day applications and gadgets. As more IoT devices are used, they handle more and more data. Managing this big data involves ensuring security and authenticity through cryptographic protocols like encryption and decryption. Traditional cryptographic methods take longer to manage the large volumes of information produced by IoT devices, especially when processing data byte by byte. These methods are seen as inefficient for handling extensive data sets. This challenge is a major obstacle in the practical implementation of IoT systems. So, there is a significant need to develop innovative, lightweight cryptographic hash functions that can address these concerns effectively. IoT devices have limitations with regard to processing, memory, and energy. Normal cryptographic hash functions are too resource-intensive for them, and most of the lightweight alternatives are not secure or efficient enough. Something that is both lightweight and highly secure has to be devised now. Most current lightweight hash functions are too complicated for small IoT devices or do not offer strong enough security. Some consume more memory, some are slow, and others are weak against attacks. No single lightweight hash function offers strong security, fast execution, and low resource consumption simultaneously. Therefore, there is a very evident need for a fresh hash function that performs well on devices with limited resources and maintains high security and efficiency for IoT purposes. A hash function is a mathematical operation that converts an input message of any size into an output message of a predetermined size. This technique finds applications in tasks like ensuring data integrity, creating Message Authentication Codes (MACs), and generating digital signatures. The primary attribute of a non-colliding hash function is that it maps inputs of varying lengths to fixed-length strings, often referred to as fingerprints [1, 2]. To thwart attackers, the objective is to make computations intricate, preventing collisions or secondary breaches. In the case of an ideal n-bit hash function, attackers require 2n/2 and 2n calculations, respectively, to find collisions or secondary breaches.

The construction and compression functions are the two important parts of a hash function. The construction function's task is to replicate the compression function's behaviors. While a random oracle might not exist, the hash function's design must align with established security criteria. An effective cryptographic hash function is anticipated to possess the subsequent key security attributes [3]:

- Collision-resistance: Breaking the collision-resistance of the hash function involves a significant challenge – finding two distinct messages, m0 and m1, for which their hash values satisfy H (m0) = H (m1). This activity necessitates a minimum computing effort of 2n/2.

- Preimage-resistance: Understanding the preimage resistance feature requires substantial effort. The aim involves identifying the initial message m from a particular H(m) hash value. This project requires a minimum of 2n computational operations.
- Second preimage-resistance: Breaking the second preimage-resistance barrier is a significant challenge. Given an initial input m0, the objective is to identify a different input m1 so that their hash values match: H(m0) = H(m1). This endeavour necessitates a computational workload of at least 2^n units.

This study aims to develop a new, lightweight cryptographic hash function known as Tiny Hash Function (THF). This new design aims to produce a hashing mechanism that follows the recognized security standards of regular hash functions and meets the special needs of lightweight applications. The proposed THF aims to provide robust security and maximum efficiency targeted for lightweight applications. The paper is organized as follows: Within Section 2, a literature review of lightweight cryptographic hash algorithms is offered. Section 3 introduces the unique lightweight hash function that is being suggested. Section 4 provides a quick description of the THF-related security analysis findings. Lastly, the research article is concluded in Section 5.

2. Literature Review

Explaining the research, Hash functions use different structures to achieve cryptographic purposes, such as Merkle-Damgård [4] and Sponge construction [5]. Merkle and Damgård independently discovered the Merkle-Damgård structure, the foundation of cryptographic hash functions from their early days, in 1989. The MD iterative method is used in this design, a compression function that accepts an input length value and also creates constant-sized output values. The Sponge structure works in a novel way: it absorbs an input message of arbitrary length into an internal state that begins as a fixed-size buffer. It then repeatedly applies to the internal state a permutation function, a process known as “absorbing”. After absorbing the incoming data, the Sponge construction enters the “squeezing” Phase, continuously pulling fixed-size output blocks from the internal state. The squeezing Phase can provide any output data, making the architecture adaptable to various applications.

Because of their large memory footprint and high power consumption, traditional hash algorithms may not be suitable for scenarios requiring limited space. As a result, the following lightweight hash functions have evolved. PHOTON, developed by Guo et al. [28], combines a sponge-like structure with an AES-style internal permutation to provide 64-bit collision resistance. It is compact at 1120 gate equivalents and provides customizable output (64 to 256 bits) and a changeable hash digest (100 to 288 bits) via parameters r, r’, and n. PHOTON incorporates a lightweight block cipher suited for hardware and software settings. It includes the PRESENT S-box, a 4-bit to 4-bit mapping that increases encryption complexity and nonlinearity. This S-box, built into PRESENT, dramatically improves cipher security and efficiency, making it appropriate for resource-constrained scenarios that require lightweight cryptographic solutions. QUARK, a lightweight hash function introduced by Aumasson in 2010, utilizes the sponge construction technique [27], emphasizing memory efficiency. Its core is a permutation, P, inspired by M. Hell [6] and C. De Canniere [7]. Offering three variants (U, D, T Quarks) for 64, 80, 112-bit hash values, QUARK ensures minimum security levels, requiring 1379 to 2296 Gate Equivalents (GE) [27].

KECCAK, a lightweight hash function devised by Kavun and Yalcin [8], features 200 and 400 variants in SHA-3. Rooted in the sponge construction, it employs the Keccak-f permutation with selectable width ‘b’ from 25 to 1600. Renowned for high-speed performance, robust security, and resilience, Keccak has undergone rigorous scrutiny, earning high-speed performance, robust security, and robustness against potential vulnerabilities. BLAKE, by Jean-Philippe Aumasson [9], comprises cryptographic hash functions like BLAKE2s, BLAKE-256, BLAKE2b, and BLAKE-512, indicating output size. With a parallel design and broad cryptographic permutation, it uses the Merkle-Damgård construction to process data in blocks. Recognized for security, simplicity, and performance, BLAKE undergoes extensive analysis and standardization, finding widespread use across diverse applications. The Gimli function, developed by Bernstein et al. [33, 10], employs a single permutation for authenticated encryption and hashing tasks. It utilizes a 384-bit state arranged in a 3x4x32 matrix, undergoing 24 nonlinear and linear operations rounds. Notably, every second and fourth round includes linear mixing and constant addition. A 96-bit SP-box functions on every state matrix column in the nonlinear layer to ensure nonlinearity. Swap operations, including big and small swaps, manipulate rows in the linear layer. Round indices 1 to 24 are utilized, with round constants XORed at multiples of four, resulting in a 32-byte hash output. Lesamnta-LWT is a hash function leveraging AES as its core component, employing Merkle-Damgård construction. The chief objective of this digest is to engender a hash output of 256 bits, which stands out with an impressive security strength of 2120, addressing collision, preimage, and second preimage resistance. Its creators substantiate its efficacy in ensuring robust cryptographic properties [25, 11].

Bussi et al. introduced a novel hash function named Neeva-hash [32], specifically designed for lightweight cryptography applications. This innovative approach draws its foundation from the sponge mode of operation and, notably, leverages the well-established sponge construction utilized in the victorious Secure Hash Algorithm-3. On this hash, the input message undergoes XOR operations with the state matrix observed by means of five rounds of compression feature in its absorption segment, a sequence of transitions has been incorporated within the squeezing section to generate the desired 224-bit output hash. Mukundan et al. [12, 13] introduced a hash based on sponge construction. Similar to the prior approaches, Hash-One incorporates the essential absorb and squeeze phases within its design framework. Notably, Hash-One is designed to generate a consistent 160-bit hash output as a result of the complex operations conducted during the squeezing Phase, which includes permutation phases.Z. A. Al-Odat et al. [14] presented a novel lightweight cryptographic hash function that uses S-Box, linear transformation, and bit permutation operations. Their suggested framework underwent thorough testing and validation, with a focus on security analysis. A detailed security analysis shows that their method successfully meets the key security requirements of common hash protocols.

Nabeel et al. [15] proposed a fresh and new hash function called LNMNT Hash, exclusively for resource-constrained settings. This unique hash underwent a thorough examination process, demonstrating its ability across multiple dimensions. Rigorous testing included going through the NIST test suite to evaluate randomness, diffusion, confusion, and susceptibility to various attack types. S. Windarta et al. [16] focused on the critical importance of Lightweight Cryptographic Hash Functions (LWCHFs) in accelerating IoT progress. The researchers clearly described the current design trends, cryptographic properties, and the range of cryptanalytic attacks associated with advanced lightweight cryptographic hash functions designed primarily for deployment in extremely limited devices. Their contribution also included a comparative analysis of several implementations in the hardware and software sectors. While numerous lightweight cryptography hash functions have been investigated, the need for fresh techniques is clear in view of rising security issues and the growth of IoT devices. Continued research is critical for developing novel solutions capable of tackling emerging difficulties and ensuring the security of their technology in the face of global dynamic change. Windarta et al. [17] designed a pair of efficient hash algorithms tailored for IoT devices, named ALIT and TJUILIK, which are built using the SATURNIN cipher and Beetle mode. Both ALIT-Hash and TJUILIK-Hash have strong security and were typically shown to perform similarly or better than existing functions in hardware and simulation testing. The authors showed that the pseudo-random values from both functions passed all NIST randomness tests. However, Windarta et al. did not do side-channel or post-quantum analyses on either function. Overall, Windarta et al. concluded that both hash functions were secure, efficient, and affiliated with IoT applications because the authors both balanced performance and resource consumption. Overall, this paper contributes to the field of lightweight cryptographic solutions for constrained environments.

The IoT enables physical objects to create, receive, and share data with minimal human involvement, aiming to increase automation, comfort, and efficiency. However, ensuring security, privacy, and trust becomes critical as IoT applications grow. This paper discusses the key security challenges and threats present in IoT systems and emphasizes the need for architectural changes to provide end-to-end security; it also addresses how new technologies like blockchain, edge computing, fog computing, and machine learning can help with the security of IoT environments and provide means of establishing trust [18].

3. Proposed Methodology

Designing lightweight cryptographic algorithms is a complex task that involves finding a balance between resource constraints and maintaining a high level of security. Many lightweight hashing approaches make the mistake of oversimplifying existing cryptographic techniques, assuming that being "lightweight" means compromising security. This is not the case, as lightweight designs must still provide robust security comparable to traditional cryptography. To address this challenge, a new lightweight cryptographic hash function, Tiny Hash Function (THF), has been developed.

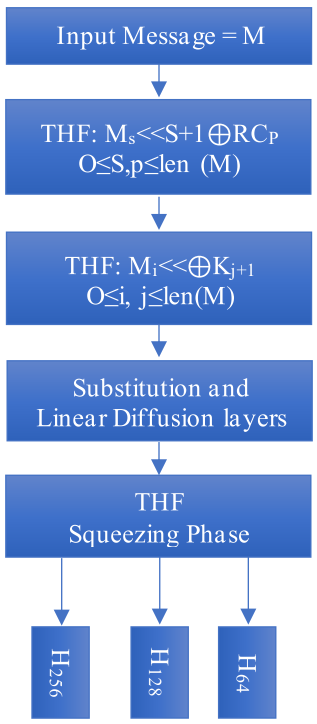


Fig. 1 Proposed system architecture

3.1. Proposed Architecture

The architecture of THF utilizes the sponge construction technique, which allows for minimizing the internal memory size while maintaining desirable security. THF is specifically tailored for efficient operation, processing an input message of any length to produce 64H/128H /256H hex hash values (i.e., 256/512/1024 bits) as illustrated in Figure 1. The proposed THF design involves multiple well-defined phases, viz., Processing the Message Blocks, Compression functions, Substitution and Linear Diffusion layers (absorption phase), Squeezing Phase aimed at achieving the desired robust security.

3.1.1. Processing the Message Blocks

The message processing in THF includes processing functional operations on 64-bit blocks of the input message. For each block, a 64-bit round constant is used, generated with the help of random prime numbers. Prime numbers are chosen due to their unique properties, such as difficulty in factorization and their significance in cryptographic operations, making them fundamental for security.

3.1.2. Compression Functions

THF uses a compression function to mix the message blocks and keys, employing bitwise XOR operations, modular additions, and rotations. It carries out 8 rounds of compression to thoroughly blend the message blocks with the keys.

3.1.3. Substitution and Linear Diffusion Layer

The Substitution layer in THF introduces confusion into the state of the sponge function, enhancing security. It is also known as the absorption phase of THF. It employs a fixed-size substitution box (S-box) that operates on portions of the keys, designed with features of cryptography like nonlinearity and resistance to attacks. However, the Linear Diffusion layer in THF introduces diffusion into the state, ensuring that changes in input affect multiple output bits. It uses linear operations like bitwise XOR and rotations to spread the influence of each input bit, making the output statistically unrelated to the input and thwarting pattern detection by potential attackers. Together, these processes form the core of THF's cryptographic sponge function, providing strong security for various applications in the IoT world.

3.2. THF-Implementation and Experimental Setup

The predicted hash algorithm is conducted on the Linux platform using an Intel(R) Core(TM) i3-6100U CPU @ 2.30GHz and 4GB of RAM.

3.2.1. Algorithm for THF

The goal is to create 64/128/256 hexadecimal hash values from an input message 'M' of varied lengths. Begin by initializing important blocks with confidential values.

- Divide the input message 'M' into smaller chunks.
- Generate a set of confidential Round Constants (RC) and keep them in a list.
- Each message block undergoes shifting operations and is combined through XOR with the respective Round Constant (RC). This processed block then enters the absorption phase.
- During the absorption phase, a thorough blending of message blocks and key blocks is executed. This involves intricate bitwise XOR operations, modular additions, and rotations to achieve a comprehensive mixing effect. Execute steps 3 and 4 for all message blocks in the input
- The result derived from the absorption phase is funnelled into the squeezing Phase to produce the desired hash output of specified length, ultimately

Numerous concealed values are initialized within the absorption phase of the THF computation. These values remain concealed from external visibility, rendering reversing the absorption phase notably challenging.

4. Security Analysis and the Results

This section assesses the resistance of the THF to a range of cryptographic attacks. This involves a comprehensive investigation into the even distribution of the hash digest, its ability to resist collisions, preimage vulnerability, and resistance to second preimage attacks, all of which contribute to evaluating the hash function's level of security.

4.1. Uniform Distribution of THF

Uniform distribution can be treated as one of the fundamental attributes of a hash. To comprehend this trait, illustrate it by envisioning a set of messages, namely M1, M2, M3, M4, and graphing them on a two-dimensional plot. Figure 2 shows that the plain text representation of the sample messages M1, M2, M3, and M4 spreads across a set of ASCII values, typically ranging from 32 to 126. This range covers various printable characters. On the other hand, Figure 3 presents the hexadecimal representation of the hash digest generated by THF. It becomes evident that the THF exhibits a uniform and random distribution of values. This uniformity ensures that the proposed hash function produces vastly different outputs for slightly different inputs, promoting collision resistance. Moreover, the hash function's uniform distribution effectively conceals statistical patterns or information from the original plain text. This property is vital to prevent attackers from deriving insights about the input data from the hash digest, thereby bolstering the overall security of the hash function.

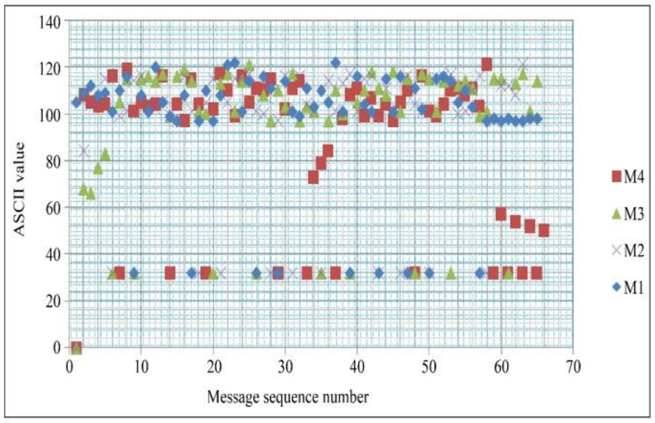


Fig. 2 Distribution of messages

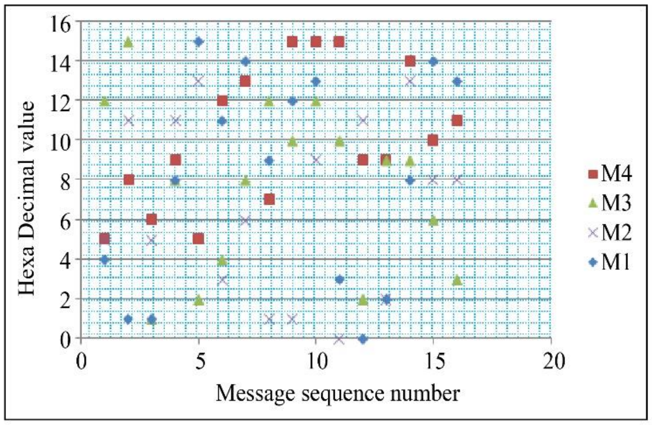


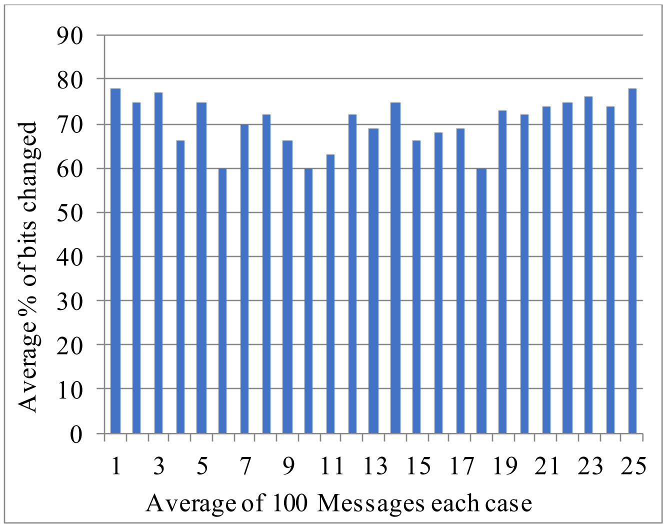
Fig. 3 Distribution of hash digests

4.2. Diffusion and Confusion Analysis

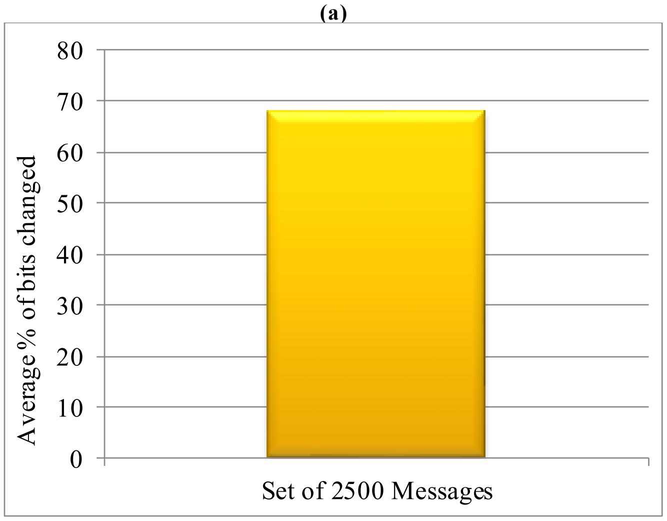
In the context of hash function design, the terms "diffusion" and "confusion" are two fundamental concepts introduced by Claude Shannon [19], and prime numbers play a significant role in achieving both. Diffusion entails a property that ensures even a minor alteration in the message leads to a significant modification in the hash digest. To achieve effective diffusion in a cryptographic hash function, a slight change in the message should lead to a significant alteration, i.e., approximately half of the final hash. On the other hand, confusion refers to a characteristic that aims to complicate and obscure the connection between the message and the hash digest.

Including prime numbers in the design of the THF serves the purpose of achieving a desirable level of confusion and diffusion. To validate this claim, an experiment is conducted as follows: An initial step involves selecting a random message and generating its hash value using the THF. So, a single bit within the message is toggled randomly, leading to a marginally modified message, and a similar hash value is recalculated. A detailed comparison is carried out between the two hash values, evaluating the number of changed bits. To ensure statistical significance, this process is repeated 2500 times, involving different randomly selected messages and bit toggling for each iteration. For every set of 100 messages, the average percentage of changed bits in the hash values is calculated. This data is then plotted in Figure 4, illustrating the graph representing the average percentage of altered bits in the hash values for the tested messages, in which Figure 4(a) shows the Average % of bits changed for every 100 messages in each case. The experimental results shown in Figure 4(b) help illustrate the diffusion level achieved by the THF.

If the average percentage of changed bits is close to 50%, it indicates that even minor adjustments to the input messages lead to widespread and balanced modifications in the corresponding hash values. In this experimental analysis, this value is 70.2%. This property validates the effectiveness of the THF in uniformly distributing the influence of individual message bits across the entire hash output, making it highly resistant to attacks seeking to deduce patterns or relationships between the input and the hash digest.



(a)



(b)

Fig. 4 Average percentage of altered bits in the hash values for the tested messages. (a) Average % of bits changed for every 100 messages in each case, and (b) Average % of bits changed for 2500 messages.

4.3. Collision Resistance

This property presents a substantial hurdle: discovering two separate messages, m0 and m1, that result in hash values meeting the condition H(m0) = H(m1). Accomplishing this task requires a computational effort of at least 2n/2.To assess the collision resistance of THF, it undergoes a rigorous test. A message was randomly selected for this test, and its corresponding hash value was generated. To challenge the hash function's collision resistance, random modifications were made to a part of the same message and recalculated its hash value. A meticulous comparison is performed for a pair of hash values, meticulously tallying the number of identical ASCII values situated at corresponding positions.

This collision resistance test was repeated N times. Out of the N test cases, 86% showed no identical characters in corresponding positions within the hash values, indicating a high level of uniqueness and resistance to collisions. There were 9% cases where only one character was found to be identical at the same location, signifying a minor collision level but still maintaining a strong overall collision resistance.

Furthermore, the examination disclosed that approximately 4% of cases exhibited two identical characters appearing in the hash values at corresponding positions. Moreover, the highest recorded occurrences of three identical characters in the same position amounted to just 1%, a notably minimal figure. These findings underscore the hash function's efficacy in mitigating such collision occurrences. Drawing from the outcomes of this experiment, it is justifiable to assert that our proposed hash function demonstrates a markedly low likelihood of collisions, which implies its robustness against second-preimage resistance.

This validates the collision resistance of the THF, transforming it into a reliable choice for applications where data integrity and security are paramount concerns. Preimage resistance states that cracking a message m should be quite hard, though H(m) is given, which needs at least 2n work. On the other hand, 2500 unique hash values were generated for the same message m, and THF is designed to support variable parameters. From this, it can be deduced that revealing the hash H(m) does not make it straightforward to decipher the original message m.

4.4. Comparative Security Analysis of THF

Preimage resistance, second preimage resistance, and collision resistance constitute essential attributes within cryptographic hash functions. These characteristics specify the degree of security and defense against different attacks a hash function should have.

A comprehensive assessment of the THF's security is presented in Table 1, comparing it to several established lightweight cryptographic hash functions. Preimage, second preimage and collision resistance values depicted in the last row of Table 1 are evident that the proposed THF can withstand desirable cryptography attacks.

Table 1. Security analysis of THF and existing lightweight cryptography hash functions

Algorithm	Hash Value	Construction	Preimage resistance	2ndPreimageresistance	Collision-resistance
Merkle-Damgård Construction: ARMADILLO & ARMADILLO2 [20]	80	Data-dependent bit transposition [56]	2^{80}	2^{80}	2^{40}
	128		2^{128}	2^{128}	2^{64}
	160		2^{160}	2^{160}	2^{80}
	192		2^{192}	2^{192}	2^{96}
	256		2^{256}	2^{256}	2^{18}
AI-Odat et al.LWCHF [21]	160	JH mode	2^{160}	2^{160}	2^{80}
	224		2^{224}	2^{224}	2^{112}
	256		2^{256}	2^{256}	2^{128}
	384		2^{384}	2^{384}	2^{192}
	512		2^{512}	2^{512}	2^{256}
EI Hanouti et al.LWCHF[22]	128	Feistel-like structure; skew tent map Davies-Meyer	2^{128}	2^{128}	2^{64}
Block Cipher-Based Construction: DM-PRESENT	64		2^{64}	None	None
	64				
H-PRESENT [23, 24]	128	Hirose construction[54]	2^{128}	None	None
C-PRESENT	192		2^{192}	None	None
Lesamnta-LW [25]	256	LW I block cipher	2^{256}	2^{256}	2^{120}
TWISH [26]	128	Davies-Meyer	2^{128}	2^{128}	2^{64}
Sponge Construction: QUARK [27]	136	P-sponge	2^{128}	2^{64}	2^{64}
	176		2^{160}	2^{80}	2^{80}
	256		2^{224}	2^{112}	2^{112}
PHOTON [28]	64	P-sponge	2^{64}	2^{40}	2^{40}
	80		2^{112}	2^{80}	2^{80}
	128		2124	2^{64}	2^{64}
	224		2^{192}	2^{112}	2^{112}
	256		2^{224}	2^{128}	2^{128}
SPONGENT [29]	80	P-sponge	2^{80}	2^{40}	2^{40}
	128		2^{120}	2^{64}	2^{64}

	160		2^{144}	2^{80}	2^{80}
	224		2^{208}	2^{112}	2^{112}
	256		2^{240}	2^{128}	2^{128}
	128	P-sponge	2^{128}	?	2^{64}
	256	JH mode	2^{128}	?	2^{64}
GLUON [30]	128	T-sponge	2^{128}	2^{64}	2^{64}
	160		2^{160}	2^{80}	2^{80}
	224		2^{224}	2^{112}	2^{112}
SipHash	64	T-Sponge JH mode	2^{64}	2^{64}	none
LHash [31, 32]	80	P-sponge	2^{64}	2^{40}	2^{40}
	96		2^{80}	2^{40}	2^{40}
	128		2^{96}	2^{56}	2^{56}
	128		2^{120}	2^{60}	2^{60}
Neeva-hash [33]	256	P-sponge; ARX	2^{224}	2^{112}	2^{112}
Gimli-Hash [34, 35]	256	P-sponge	2^{128}	2^{128}	2^{128}
sLiSCP-hash [35, 36, 37]	160	P-sponge	2^{128}	2^{80}	2^{80}
	192		2^{128}	2^{96}	2^{96}
	192		2^{160}	2^{96}	2^{96}
LNhash [38]	80	P-sponge	2^{72}	2^{40}	2^{40}
	96		2^{80}	2^{40}	2^{40}
	128		2^{96}	2^{56}	2^{56}
	128		2^{120}	2^{60}	2^{60}
	160		2^{144}	2^{80}	2^{80}
	160		2^{152}	2^{80}	2^{80}
ACE (ACE-H-256) [39]	256	P-sponge	2^{192}	2^{128}	2^{128}
ASCON-HASH [40]	256	P-sponge	2^{128}	2^{128}	2^{128}
KNOT-hash [41]	256	P-sponge	2^{128}	2^{112}	2^{112}
	256		2^{128}	2^{128}	2^{128}
	384		2^{192}	2^{168}	2^{168}
	512		2^{256}	2^{224}	2^{224}
DryGASCON [42]	128	DrySponge	None	None	2^{64}
	256	P-sponge	None	None	2^{128}
ORANGISH [43]	128	P-sponge	2^{128}	2^{112}	2^{112}
PHOTON_Beetle-Hash [44]	128	P-sponge	2^{128}	2^{112}	2^{112}
ESCH [45]	256	P-sponge	2^{128}	2^{128}	2^{128}
	384		2^{192}	2^{192}	2^{192}
Subterranean 2.0 [46]	256	P-sponge	2^{224}	2^{224}	2^{224}
Xoodyak Hash Mode [47]	256	P-sponge	2^{128}	2^{128}	2^{128}
HVH [48]	88	P-sponge	2^{72}	2^{10}	2^{40}
	128		2^{120}	2^{64}	2^{64}
	160		2^{144}	2^{80}	2^{80}
	224		2^{208}	2^{112}	2^{112}
	256		2^{224}	2^{128}	2^{125}
LNMNT Hash [49]	80	P-sponge	2^{50}	-	-
	128		2^{80}	-	-
	160		2^{100}	-	-
	224		2^{120}	-	-
Cellular Automata: L-CAHASH	128	Cellular Automata	2^{128}	2^{128}	2^{64}
	256		-	-	-
LCAHASH1.1	128	Cellular Automata	2^{128}	2^{128}	2^{64}
	256		-	-	-

THF (Proposed one)	**(64)H**	**Sponge**	2^{256}	2^{256}	2^{128}
	(128)H		2^{512}	2^{512}	2^{256}
	(256)H		2^{1024}	2^{1024}	2^{512}

5. Conclusion

This research article aims to develop a new lightweight cryptographic hash function, i.e., Tiny Hash Function, termed THF. The proposed design employs Processing the Message Blocks, Compression functions, Substitution and Linear Diffusion layers, and Squeezing Phase. THF had a primary focus on three crucial lightweight requirements: "collision resistance, preimage resistance, and second preimage resistance", and the results are evident that it can withstand desirable cryptography attacks. The proposed design may be evaluated on various energy resource parameters to support resource-constrained applications.

References

[1] Atul Kahate, *Cryptography and Network Security*, McGraw-Hill International ed., 2003. [Google Scholar]

[2] Mohammad Reza Sohizadeh Abyaneh, "*Security Analysis of Lightweight Schemes for RFID Systems*," Doctoral Thesis, The University of Bergen, pp. 1-162, 2012. [Google Scholar] [Publisher Link]

[3] Elena Andreeva, Bart Mennink, and Bart Preneel, "Security Properties of Domain Extenders for Cryptographic Hash Functions," *Journal of Information Processing Systems*, vol. 6, no. 4, pp. 453-480, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[4] Ivan Bjerre Damgård, "A Design Principle for Hash Functions," *Conference on the Theory and Application of Cryptology*, Springer New York, pp. 416-427, 1989. [CrossRef] [Google Scholar] [Publisher Link]

[5] Guido Berton et al., "Sponge Functions," *In ECRYPT Hash Workshop*, vol. 2007, no. 9, pp. 1-23, 2007. [Google Scholar] [Publisher Link]

[6] Martin Hell, Thomas Johansson, and Willi Meier, "Grain: A Stream Cipher for Constrained Environments," *International Journal of Wireless and Mobile Computing*, vol. 2, no. 1, pp. 86-93, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[7] Christophe De Cannière, Orr Dunkelman, and Miroslav Knežević, "KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block *Ciphers*," *Cryptographic Hardware and Embedded Systems - CHES 2009 11th International Workshop Lausanne,* Switzerland, pp. 272-288, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[8] Elif Bilge Kavun, and Tolga Yalcin, "A Lightweight Implementation of Keccak Hash Function for Radio-Frequency Identification Applications," *Radio Frequency Identification: Security and Privacy Issues 6th International Workshop, RFIDSec 2010*, Istanbul, Turkey, pp. 258-269, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[9] Jean-Philippe Aumasson et al., *The Hash Function BLAKE*, 1st ed., Information Security and Cryptography, Springer Berlin, Heidelberg, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[10] George Hatzivasilis et al., "A Review of Lightweight Block Ciphers," *Journal of Cryptographic Engineering*, vol. 8, no. 2, pp. 141-184, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[11] Asraf Akhimullah, and Shoichi Hirose, "Lightweight Hashing Using Lesamnta-LW Compression Function Mode and MDP Domain Extension," *2016 Fourth International Symposium on Computing and Networking (CANDAR)*, Hiroshima, Japan, pp. 590-596, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[12] Puliparambil Megha Mukundan et al., "Hash-One: A Lightweight Cryptographic Hash Function," *IET Information Security*, vol. 10, no. 5, pp. 225-231, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[13] K. Shankar, and Mohamed Elhoseny, *An Optimal Lightweight Cryptographic Hash Function for Secure Image Transmission in Wireless Sensor Networks*, Secure Image Transmission in Wireless Sensor Network (WSN) Applications, Springer, Cham, pp. 49-64, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[14] Zeyad A. Al-Odat, Eman M. Al-Qtiemat, and Samee U. Khan, "An Efficient Lightweight Cryptography Hash Function for Big Data and IoT Applications," *2020 IEEE Cloud Summit*, Harrisburg, PA, USA, pp. 66-71, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[15] Nubila Nabeel, Mohamed Hadi Habaebi, and M.D. Rafiqul Islam, "Security Analysis of LNMNT-LightWeight Crypto Hash Function for IoT," *IEEE Access*, vol. 9, pp. 165754-165765, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[16] Susila Windarta et al., "Lightweight Cryptographic Hash Functions: Design Trends, Comparative Study, and Future Directions," *IEEE Access*, vol. 10, pp. 82272-82294, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[17] Susila Windarta et al., "Two New Lightweight Cryptographic Hash Functions Based on Saturnin and Beetle for the Internet of Things," *IEEE Access*, vol. 11, pp. 84074-84090, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[18] Vikas Hassija et al., "A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures," *IEEE Access*, vol. 7, pp. 82721-82743, 2019.[CrossRef] [Google Scholar] [Publisher Link]

[19] C.E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379-423, 1948. [CrossRef] [Google Scholar] [Publisher Link]

[20] Stéphane Badel et al., "ARMADILLO: A Multi-Purpose Cryptographic Primitive Dedicated to Hardware," *Cryptographic Hardware and Embedded Systems -- CHES 2010 12th International Workshop*, Santa Barbara, USA, pp. 398-412, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[21] Bruno O. Bracht et al., "Data Authentication using Modification Detection Codes Based on a Public One Way Encryption Function," *U.S. Patent 4908861A*, pp. 1-29, 1990. [Google Scholar] [Publisher Link]

[22] Imad El Hanouti et al., "A Lightweight Hash Function for Cryptographic and Pseudo-Cryptographic Applications," *WITS 2020 Proceedings of the 6th International Conference on Wireless Technologies, Embedded, and Intelligent Systems*, Singapore, pp. 495-505, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[23] Andrey Bogdanov et al., "Hash Functions and RFID Tags: Mind the Gap," *Cryptographic Hardware and Embedded Systems - CHES 2008 10th International Workshop*, Washington, DC, USA, pp. 283-299, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[24] Axel York Poschmann, *Lightweight Cryptography-Cryptographic Engineering for a Pervasive World*, Dissertation, Ruhr-University Bochum, Germany, pp. 1-197, 2009. [Google Scholar] [Publisher Link]

[25] Shoichi Hirose et al., "A Lightweight 256-Bit Hash Function for Hardware and Low-End Devices: Lesamnta-LW," *Information Security and Cryptology - ICISC 2010 13th International Conference*, Seoul, Korea, pp. 151-168, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[26] Deden Irfan Afryansyah, Magfirawaty, and Kalamullah Ramli, "The Development and Analysis of Twish: A Lightweight-Block-Cipher-Twine-Based Hash Function," *2018 Thirteenth International Conference on Digital Information Management (ICDIM)*, Berlin, Germany, pp. 210-215, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[27] Jean-Philippe Aumasson et al., "Quark: A Lightweight Hash," *Journal of Cryptology*, 2013, vol. 26, no. 2, pp. 313-339, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[28] Jian Guo, Thomas Peyrin, and Axel Poschmann, "The PHOTON Family of Lightweight Hash Functions," *Advances in Cryptology-CRYPTO 2011 31st Annual Cryptology Conference*, Santa Barbara, CA, USA, pp. 222-239, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[29] Andrey Bogdanov et al., "SPONGENT: A Lightweight Hash Function," *Cryptographic Hardware and Embedded Systems--CHES 2011 13th International Workshop*, Nara, Japan, pp. 312-325, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[30] Thierry P. Berger et al., "The GLUON Family: A Lightweight Hash Function Family Based on FCSRs," *Progress in Cryptology--AFRICACRYPT 2012 5th International Conference on Cryptology in Africa*, Ifrane, Morocco, pp. 306-323, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[31] Wenling Wu et al., "LHASH: A Lightweight Hash Function," *Information Security and Cryptology 9th International Conference, Inscrypt 2013*, Guangzhou, China, pp. 291-308, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[32] Khushboo Bussi et al., "Neeva: A Lightweight Hash Function," *IACR Cryptology ePrint Archive*, 2016. [Google Scholar] [Publisher Link]

[33] Daniel J. Bernstein et al., "GIMLI: A Cross-Platform Permutation," *Cryptographic Hardware and Embedded Systems - CHES 2017 19th International Conference*, Taipei, Taiwan, pp. 299-320, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[34] Riham AlTawy et al., "Towards a Cryptographic Minimal Design: The sLiSCP Family of Permutations," *IEEE Transactions on Computers*, vol. 67, no. 9, pp. 1341-1358, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[35] Riham AlTawy et al., "sLiSCP: Simeck-Based Permutations for Lightweight Sponge Cryptographic Primitives," *Selected Areas in Cryptography - SAC 2017 24th International Conference*, Ottawa, ON, Canada, pp. 129-150, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[36] D.R. Stinson, "Some Observations on the Theory of Cryptographic Hash Functions," *Designs, Codes and Cryptography an International Journal*, vol. 38, no. 2, pp. 259-277, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[37] Mark Aagaard et al., "*ACE: An Authenticated Encryption and Hash Algorithm*," University of Waterloo, pp. 1-68, 2019. [Google Scholar] [Publisher Link]

[38] Christoph Dobraunig et al., Ascon V1.2, Submission to NIST, 2021. [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/ascon-spec-round2.pdf

[39] Wentao Zhang et al., "*KNOT: Algorithm Specifications and Supporting Document*," Submission to NIST Lightweight Cryptography Project, pp. 1-27, 2019. [Google Scholar] [Publisher Link]

[40] Sébastien Riou, DryGASCON Lightweight Cryptography Standardization Process Round 1 Submission, pp. 1-107, 2019. [Online]. Available: https://csrc.nist.gov/csrc/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/drygascon-spec.pdf

[41] Bishwajit Chakraborty, and Mridul Nandi, ORANGE, Isical, 2019. [Online]. Available: https://www.isical.ac.in/~lightweight/Orange/

[42] Zhenzhen Bao et al., PHOTON-Beetle Authenticated Encryption and Hash Family, pp. 1-14, 2019. [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/photon-beetle-spec-round2.pdf

[43] Christof Beierle et al., "Schwaemm and Esch: Lightweight Authenticated Encryption and Hashing Using the Sparkle Permutation Family," *NIST Round*, vol. 2, pp. 162893-162908, 2019. [Google Scholar]

[44] Joan Daemen et al., “The Subterranean 2.0 Cipher Suite,” *IACR Transactions on Symmetric Cryptology*, vol. 2020, no. S1, pp. 262-294, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[45] Joan Daemen et al., “Xoodyak, A Lightweight Cryptographic Scheme,” *IACR Transactions on Symmetric Cryptology*, vol. 2020, no. S1, pp. 60-87, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[46] Yuhua Huang et al., “HVH: A Lightweight Hash Function Based on Dual Pseudo-Random Transformation,” *Security, Privacy, and Anonymity in Computation, Communication, and Storage SpaCCS 2020 International Workshops*, Nanjing, China, pp. 492-505, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[47] Charifa Hanin et al., “L-CAHASH: A Novel Lightweight Hash Function Based on Cellular Automata for RFID,” *Ubiquitous Networking Third International Symposium, UNet 2017*, Casablanca, Morocco, pp. 287-298, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[48] Anas Sadak et al., “LCAHASH-1.1: A New Design of the LCAHASH System for IoT,” *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 11, pp. 253-257, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[49] A.A. Moldovyan, and N.A. Moldovyan, “A Cipher Based on Data- Dependent Permutations,” *Journal of Cryptology*, vol. 15, no. 1, pp. 61-72, 2002. [CrossRef] [Google Scholar] [Publisher Link]